APPLICATION FOR PATENT

INVENTOR: JIM S. KUO

TITLE: DYNAMIC LOAD BALANCING IN A MULTI-BUS COMPUTER SYSTEM

SPECIFICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] Not applicable.

STATEMENTS REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] Not applicable.

REFERENCE TO A MICROFICHE APPENDIX

[0003] Not applicable.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0004] The present invention generally relates to multi-bus computer systems and more particularly to dynamic load balancing in such systems.

2. Description of the Related Art

[0005] Most computer systems today utilize Peripheral Component Interconnect (PCI) buses as the input/output (I/O) expansion bus on which system I/O components reside. These components may include Small Computer System Interface (SCSI) controllers, network interface cards and video cards. In high performance systems such as high-end workstations or servers, there typically exist multiple PCI buses arranged in a peer bus architecture that provides the maximum number of paths directly into the memory and processors. This generally allows the highest level of concurrent access to required data and allows the best data throughput and performance. In today's system, there are several types of PCI buses and PCI-X buses available: 32-bit 33 MHz PCI, 64-bit 33 MHz PCI, 64-bit 66 MHz PCI, 100 MHz PCI-X, 133 MHz PCI-X and other types of I/O buses.

[0006] Due to the limitation of mechanical space available in a chassis of a computer system and other related factors, a system designer will have to decide on how many slots will be provided in the system and what types of slots to provide. Since different types of slots typically reside on different types of I/O buses, the system designer will have to make assumptions as to what types of cards a user will typically install in these slots. For example, the designer may have to decide whether to provide 32-bit PCI slots and/or 64-bit PCI slots and how many slots of each if both slot types are supported. Unfortunately, users tend to have different needs from computer systems. As a result, it is possible that the designer made a wrong choice in what type of slot mix to provide. In that circumstance, a user is not afforded the maximum data transfer potential of the cards. For example, the user might be forced to place a 64-bit PCI card into a 32-bit PCI slot, resulting in half the data transfer rate than if the 64-bit PCI card were in a 64-bit PCI slot. The user also may unknowingly be wasting bandwidth by heavily loading one expansion bus while another expansion bus is idle. For example, in a dual-bus computer system with three 32-bit PCI cards in 32-bit PCI slots and three 64-bit PCI cards in 64-bit PCI slots, if all three 64-bit PCI cards are assigned to one expansion bus, then a certain amount of bandwidth of the other expansion bus is being wasted.

[0007] Another reason for performance degradation is the sharing of buses by different cards. For example, if a user was unaware of the bus type of the slots and inserted a 33 MHz PCI card and a 66 MHz PCI card into the slots of the same expansion bus, the entire expansion bus will be running at the slower 33MHz speed. This reduced the performance of 66 MHz PCI cards by half. Although the current PCI standard allows backward compatibility of faster buses to accommodate the lower speeds (For example, 133MHz 64 bits PCIX bus will support 33MHz 32 bit PCI cards), it will not negotiate to each device individually. This is another reason where this invention can be important.

BRIEF SUMMARY OF THE INVENTION

[0008] A multi-bus computer system is adapted for dynamic load balancing. The computer system includes a plurality of expansion slots, a first expansion bus, a second expansion bus, and a bus switching mechanism to assign at least one of the plurality of expansion slots between the first expansion bus and the second expansion bus. Selection of the particular expansion bus may be controlled by a bus selection signal based on a type of

device in the particular expansion slot, the speed of the device in the expansion slot, the bitsize of the device in the expansion slot, or availability of each expansion slot.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0009] A better understanding of the present invention can be obtained when the following detailed description is considered in conjunction with the following drawings, in which:

Figure 1 is a block diagram of an exemplary computer system adapted for bus switching between a 32-bit PCI bus and a 64-bit PCI bus;

Figure 2 is a block diagram illustrating exemplary switch logic and signaling for dynamic assignment between the 32-bit PCI bus and the 64-bit PCI bus of Figure 1 to a 64-bit PCI slot;

Figure 3 is a truth table for the switch logic of Figure 2;

Figure 4 is a block diagram of an exemplary computer system adapted for bus switching between a 32-bit PCI bus and a 64-bit PCI bus utilizing system firmware;

Figure 5 is a flowchart of an exemplary firmware process for dynamically optimizing bus loading for a multi-bus computer system such as shown in Figure 4;

Figure 6 is an illustration in table form of exemplary stages associated with the firmware process of Figure 5;

Figure 7 is a block diagram illustrating exemplary switch logic and signaling for dynamic assignment between two 64-bit PCI buses to a 64-bit 5V PCI slot;

Figure 8 is a diagram of an exemplary architecture for dynamic switching among three PCI buses to PCI expansion slots of a computer system; and

Figure 9 is an illustration of an exemplary multiplexor architecture for dynamically switching among three PCI buses to PCI expansion slots of a computer system.

DETAILED DESCRIPTION OF THE INVENTION

[0010] Turning now to the drawings, Figure 1 shows an exemplary computer system 100 adapted for bus switching between a 32-bit Peripheral Component Interconnect (PCI) bus 108 and a 64-bit PCI bus 107 and 110. Through a host bridge 104, processors 102 are coupled to memories 106 and PCI buses 107, 108 and 110. While the upper 64 bits of the 64-bit PCI bus 107 is coupled directly to the PCI slots 114, the 32-bit PCI bus 108 and the lower 32 bits of the 64-bit PCI bus 110 are selectively coupled to the PCI slots 114 through switch logic

112. A particular slot thus may be assigned to either the 32-bit PCI bus 108 or the 64-bit PCI bus 107 and110. In this configuration, the PCI buses 107, 108 and 110 serve as expansion buses, and the slots 114 serve as expansion slots.

[0011] Referring to Figure 2, exemplary switch logic and signaling for dynamic assignment between the 32-bit PCI bus 108 and the lower 32 bits of the 64-bit PCI bus 110 to a 64-bit PCI slot 114a is shown. Either the 32-bit PCI bus 108 or the lower 32 bits of the 64-bit PCI bus 110 can be connected to the 64-bit PCI slot 114a. A portion 224 of the 64-bit PCI slot 114a is a common interface for either the 32-bit PCI bus 108 or the lower 32 bits of the 64-bit PCI bus 110, and a portion 220 of the 64-bit PCI slot 114a is only used by the upper 32 bits of the 64-bit PCI bus. The 64-bit PCI slot 114a is automatically assigned to the appropriate bus depending on the type of PCI card detected in the slot 114a. If a 64-bit PCI card is detected in the slot 114a, then the lower 32 bits of the 64-bit PCI bus 110 is assigned to the slot 114a. If a 32-bit PCI card is detected in the slot 114a, then the 32-bit PCI bus 108 is assigned to the slot 114a. The 64-bit PCI bus 218 is shown to represent the upper 32 bits of the 64-bit PCI slot 114a that may be connected to the 64-bit PCI bus because the 32-bit PCI bus does not use this portion of the slot.

[0012] In this embodiment, the switch logic 112 coupled to the 32-bit PCI bus 108 and the lower 32 bits of the 64-bit PCI bus 110 is implemented as a bus switch. The bus switch 112 may be placed as close to the 64-bit PCI slot 114a as possible. An example of a suitable bus switch is a PI5C116210 20-bit, 2-port bus switch available from Pericom Semiconductor. The bus switch 112 is shown in the form of a logic block diagram. The 32-bit PCI bus 108 serves as the input signal 1A to the source of a transistor 206. The lower 32 bits of the 64-bit PCI bus 110 is negated by an inverter 200 with the negated signal serving as a bus enable input 1OE# provided to an inverter 202. The output of the inverter 202 is provided as a control signal to the gate of the transistor 206. The 64-bit PCI bus 110 also serves as an input signal 2A to the source of a transistor 208 and as a bus enable input 2OE# negated by an inverter 204. The negated signal serves as a control signal to the gate of the transistor 208. Either an output signal 1B from the drain of the transistor 206 or an output signal 2B from the drain of the transistor 208 are provided to the PCI slot 114a depending upon the states of the transistors 206 and 208.

[0013] Turning to Figure 3, a truth table 300 corresponding to the logic block diagram of the bus switch 112 in Figure 2 is shown. The portions of the truth table 300 applicable to the logic block diagram in Figure 2 are the condition where the bus enable input 1OE# is low and the bus enable input 2OE# is high and the condition where the bus enable input 1OE# is high and the bus enable input 2OE# is low. If a 64-bit PCI card is plugged into the 64-bit PCI slot 114a, then pin A63 is grounded on the card, driving the signal 212 low. The signal 212 serves as a presence detect signal capable of detecting the presence of a 64-bit PCI card in the 64-bit PCI slot 114a. Since the signal 212 is low, the input to the inverter 200 is low and the bus enable input 1OE# is high. Since the signal 212 serves as the bus enable input 2OE#, the bus enable input 2OE# is low. According to the truth table 300, the input signal 1A and the output signal 1B are placed in a high Z or impedance state, and the output signal 2B is set to the input signal 2A. Consequently, the 32-bit PCI bus 108 is electrically isolated from the 64-bit PCI slot 114a and the lower 32 bits of the 64-bit PCI bus 110 are electrically connected to the 64-bit PCI slot 114a.

[0014] If a 32-bit PCI card is plugged into the 64-bit PCI slot 114a, then the A63 pin is pulled up by a pull-up resistor 214, driving the signal 212 high. Since the signal 212 is high, the bus enable input 2OE# is high and the bus enable signal 1OE# is low. According to the truth table 300, the input signal 2A and the output signal 2B are placed in a high Z or impedance state, and the output signal 1B is set to the input signal 1A. Consequently, the lower 32 bits of the 64-bit PCI bus 110 are electrically isolated from the 64-bit PCI slot 114a and the 32-bit PCI bus 108 is electrically connected to the 64-bit PCI slot 114a. A number of similar logical configurations to that represented in Figures 2 and 3 may alternatively be used to implement the bus switch 112.

[0015] Referring to Figure 4, an exemplary computer system 400 adapted for bus switching between the 32-bit PCI bus 108 and the lower 32 bits of the 64-bit PCI bus 110 utilizing system firmware is shown. As opposed to basing bus switching strictly upon a presence detect signal such as described in connection with Figure 2, bus switching in Figure 4 is based on logic under system firmware control. Control logic 402 enables system firmware to control the switch logic 112. Unlike the switch logic 112 in Figure 2, the switch logic 112 in Figure 4 is software controlled. A software-based approach to dynamic bus switching presents a great amount of flexibility in balancing load in a multi-bus computer system. For example, depending on the number and type of cards in the computer system

100, firmware may dynamically identify and establish a bus assignment to balance the load. With the techniques disclosed herein, maximum throughput to each available bus may be accomplished.

[0016] Referring to Figure 5, an exemplary firmware process for dynamically optimizing bus loading for a multi-bus computer system such as illustrated in Figure 4 is shown. Beginning in step 502, system power is turned on. Next, in step 504, the processors and memories in the computer system are initialized. Control then proceeds to step 506 where PCI device discovery is performed. In this step, every PCI card in a slot is identified by speed, type and bitsize. As to speed, it may determined whether the card is a 33MHz, 66 MHz, 100 MHz, or 133MHz card. Regarding type, it may be determined if the card is a PCI-card or a PCI-X card. The type of card must match the corresponding type of bus. As to bitsize, it may be determined if the card is a 32-bit card or a 64-bit card.

[0017] In step 508, the cards identified in step 506 are sorted by slot. In other words, the firmware takes account of which particular cards are found in which particular slots. From step 508, the process proceeds to step 510 where slots are grouped by common characteristics. Slots containing cards of the same speed, type and bitsize are placed in the same group. Any cards of different speed, type or bitsize are placed in different groups. The next step in the process is to determine an assignment of resources in step 512. In this step, a bus or group of buses to assign to each group of cards is determined. Next, in step 514, the firmware determines if the determined resource assignment matches the current configuration. If there is a match, the process proceeds to step 516 in which the PCI cards are initialized. Following step 516, Power On Self Test (POST) code continues to execute.

[0018] If the determined resource assignment does not match the current configuration in step 514, then the process proceeds to step 520 where the bus selection signal is set to dynamically assign the buses based on the determined resource assignment. This determined resource assignment represents a more optimized bus configuration than the current bus configuration. The state of the bus selection signal is saved in step 522. Next, in step 524, the computer system is reset to guarantee that the state of the bus selection signal is correct. From step 524, the process returns to step 504. At this point, the state of the bus selection signal is read and governs the current resource assignment until the next dynamic resource assignment.

[0019] Referring to Figure 6, an illustration in table form of exemplary stages associated with the firmware process of Figure 5 is shown. Step 602 corresponds to step 506 of Figure 5. In this example, two PCI 64-bit 66 MHz cards and two PCI-X 64-bit 100 MHz cards are identified during PCI device discovery. Stage 604 corresponds to step 508 of Figure 5. Slot_1 is identified as containing a PCI 64-bit 66 MHz card; Slot_2 is identified as containing a PCI-X 64-bit 100 MHz card; Slot_3 is identified as containing a PCI 64-bit 66 MHz card; and Slot_4 is identified as containing a PCI-X 64-bit 100 MHz card. Stage 606 corresponds to step 510 of Figure 5. Slot_1 and Slot_3 are grouped as containing PCI 64-bit 66 MHz cards (Type 1). Slot_2 and Slot_4 are grouped as containing PCI-X 64-bit 100 MHz cards (Type 2). Stage 608 corresponds to step 512 of Figure 5. In this example, the optimal resource assignment is to assign Bus_1 to Slot_1 and Slot_3 and to assign Bus_2 to Slot_2 and Slot_4. Stage 610 corresponds to step 520 of Figure 5. The bus selection signal sets Slot_1 to Bus_1, Slot_2 to Bus_2, Slot_3 to Bus_1, and Slot_4 to Bus_2.

[0020] Referring to Figure 7, exemplary switch logic and signaling for dynamic assignment between a 64-bit PCI bus 702 and a 64-bit PCI bus 704 to a 64-bit PCI slot 726 is shown. A software programmable signal 728 is utilized in connection with a bus switch 112 to control bus switching for the 64-bit PCI buses 702 and 704. A suitable example of a software programmable signal is a general purpose input/output (GPIO) signal. The bits of the GPIO signal may be set by firmware. If the software programmable signal 728 is high, then the bus enable signal 2OE# is high and the bus enable signal 1OE# is low. By virtue of the inverter 706, the buses enable signal 1OE# is an inverted version of the software programmable signal 728. According to the truth table 300, the input signal 2A and the output signal 2B are placed in a high Z or impedance state, and the output signal 1B is set to the input signal 1A. Consequently, the 64-bit PCI bus 704 is electrically isolated from the 64-bit PCI slot 726 and the 64-bit PCI bus 702 is electrically connected to the 64-bit PCI slot 726.

[0021] If the software programmable signal 728 is low, then the bus enable signal 2OE# is low and the bus enable signal 1OE# is high. According to the truth table 300, the input signal 1A and the output signal 1B are placed in a high Z or impedance state, and the output signal 2B is set to the input signal 2A. Consequently, the 64-bit PCI bus 702 is electrically isolated from the 64-bit PCI slot 726, and the 64-bit PCI bus 704 is electrically connected to the 64-bit PCI slot 726. The software programmable signal 728 thus serves as a load-based

bus selection signal. While only one PCI slot is shown in Figure 7, it should be understood that the same technique may be utilized to place each slot in a computer system into the appropriate bus.

[0022] A GPIO signal 720 is coupled to an inverter 722 that connects to the 64-bit PCI slot 726 and an output line 724 from the bus switch 112. The GPIO signal 720 holds any PCI cards in the 64-bit PCI slot 726 in reset until the bus selection by the bus switch 112 is completed. In this way, the cards are prevented from accidentally driving either bus. It should be understood that the techniques disclosed may extend to buses in a computer system other than PCI buses, such as memory buses, PCI-X buses and Small Computer System Interface (SCSI) buses, for example.

[0023] Referring to Figure 8, an exemplary architecture for dynamic switching among four PCI buses to expansion slots of a computer system is shown. The architecture 800 includes four PCI buses 802, 804, 806 and 808. Switch logic 810 handles switching between PCI bus 802 and 804 to PCI slots 814; switch logic 812 handles switching between PCI buses 806 and 808 to PCI slots 814; Control signals to the switch logic will take account of bus connections and ensure only one bus is connected to PCI Slots 814. Figure 8 thus helps to show that bus switching in accordance with the disclosed techniques may extend to any number of buses.

[0024] Referring to Figure 9, an exemplary multiplexor architecture for dynamically switching among three PCI buses to expansion slots of a computer system is shown. As opposed to a bus switch, the architecture uses a multiplexor 908 to handle switching among PCI buses 902-906. Based on the particular PCI bus selected by the control signal 910, the multiplexor 908 generates an output signal 912 corresponding to the selected PCI bus. As shown, the output signal 912 is provided to PCI slots 914. The control signal 910 may be a presence detect signal such as described in connection with Figure 2 and/or a software programmable bus selection signal such as described in connection with Figure 7. Based on Figure 9, it should be understood that the multiplexor 908 may be used as an alternative to bus switch 112 in Figures 2 and 7 as long as the timing requirements are met.